UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/786,326	02/26/2004	Jun-seo Lee	Q78241	2660

23373 7590 08/05/2008

SUGHRUE MION, PLLC
2100 PENNSYLVANIA AVENUE, N.W.
SUITE 800
WASHINGTON, DC 20037

EXAMINER	
BELANI, KISHIN G	

ART UNIT	PAPER NUMBER
2143	

MAIL DATE	DELIVERY MODE
08/05/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/786,326	LEE, JUN-SEO	
	Examiner	Art Unit	
	KISHIN G. BELANI	2143	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 May 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-16* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-16* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This action is in response to Applicant's amendment filed on 05/02/2008. Only the **dependent Claims 12-14 have been amended for clarity only. Claims 1-16 are now pending** in the present application. The applicants' claim amendments are shown in ***bold and italics***, and the examiner's response to the amendments is shown in **bold** in this office action. **This Action is made FINAL**.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

Claims 1-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Crow et al. (U.S. Patent Application Publication # 2002/0161915 A1)** in view of

Ganesan et al. (U.S. Patent Application Publication # 2003/0069973 A1) and further in view of **Varma et al. (U.S. Patent Application Publication # 2004/0037302 A1)** and further in view of **Rana et al. (U.S. Patent Application Publication # 2002/0095512 A1).**

Consider **claim 1**, Crow et al. show and disclose a method for receiving a plurality of packets from a network and distributing the packets to a plurality of protocol processors (Fig.1, that shows a Border Router 16 receiving a plurality of packets from the Internet 22, and distributing the packets to a plurality of protocol processors 24; paragraph 0018 discloses the same details) comprising the steps of:

if a received packet is a fragmented packet, determining whether the received packet is a first fragment packet (Flowchart of Fig. 4, blocks 102 and 108 that test for a fragmented received packet and then check if it is the first (primary) fragment packet; paragraph 0037, lines 1-5 that disclose testing for a fragmented packet; paragraph 0038 that discloses a test for the fragmented received packet being the first such packet);

searching an index indicating one of the protocol processors (Fig. 1, Translation Table 82 with entries in it; Fig. 3 that shows an entry matching the IP and Transport Header information (shown in Fig. 2A) in the primary fragment, after searching the translation table 82 for a matching protocol processor);

entering the index into the corresponding list (Fig. 3, a group of Fragment Contexts shown below Fragment Context 92 and representing a list, for the secondary segments that were previously stored, being updated with the fragment context of the primary

fragment; Fig. 4, blocks 114-116; paragraphs 0039-0040 that describe generating a fragment-context 92 (shown in Fig. 3) for the identified translation entry and applying it to the secondary fragments); and

attaching the index as a tag to the received packet and transmitting the received packet to the corresponding one of the protocol processors (Fig. 4, block 116, paragraph 0039 that discloses the translation and subsequent transmission process for the primary fragment).

However, Crow et al. do not disclose a method wherein if the received packet is the first fragment packet, looking up a tunnel ID of the received packet from a tunnel ID look-up table; looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list; and if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table.

In the same field of endeavor, Ganesan et al. disclose a method for looking up a tunnel ID of the received packet from a tunnel ID look-up table (Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing looking up a tunnel ID of the received packet from a tunnel ID look-up table).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method for looking up a tunnel ID of the received packet from a tunnel ID look-up table, as taught by Ganesan et al., in the method of Crow et al., so that encapsulated received packets can be securely delivered through the firewall of the receiving node.

However, Crow et al., as modified by Ganesan et al., do not specifically disclose a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list; and if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table.

In the same field of endeavor, Varma et al. disclose a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list (Fig. 3; that shows Data Memory 210 for storing packets, Control Memory 200 for storing the fragment look-up table with head and tail pointers and other data (including packet count) for each list, and Link Memory 220 with pointers to keep track of the related packets of the list in the data memory; paragraph 0032, lines 6-13 which disclose that a determination

is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the packet; by checking the packet count value for zero. If the packet count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing a method wherein if the received packet is the first fragment packet, looking-up a fragment ID (queue id) of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list, as taught by Varma et al., in the method of Crow et al., as modified by Ganesan et al., so that the related packets received in the out of order sequence, may be organized in a single list for subsequent translation and transmission to a common destination host/port.

However, Crow et al., as modified by Ganesan et al. and Varma et al., do not specifically disclose a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the

corresponding list of the fragment look-up table. Although Crow et al. do disclose creating a fragment context for a primary fragment and associating it with the related secondary fragments for subsequent translation and transmission of the related secondary fragments, Crow et al. do not specifically associate it with a list in the fragment look-up table.

In the same field of endeavor, Rana et al. disclose a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table (Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing link list, as discussed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table, as taught by Rana et al., in the method of Crow et al., as modified by Ganesan et al. and Varma et al., so that the related packets received in the out of order sequence, may be organized in a single list for subsequent translation and transmission to a common destination host/port.

Consider **claim 2**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose a method, wherein the step of entering the index into the corresponding list of the fragment look-up table, includes newly entering the result of the looked-up fragment ID and the index into the fragment look-up table, if the list corresponding to the result of the looked-up fragment ID does not exist in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the data; by checking the packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thereby disclosing creating a new list with the fragment id (queue id), if the list corresponding to the result of the looked-up fragment ID does not exist in the fragment look-up table); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry; Flowchart of Fig. 4, blocks 110-114; paragraph 0040 that discloses the process of generating an index (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, thereby disclosing generating an index); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a new fragment, in which case a fragment id (index) is assigned to the data packet); together disclosing a method wherein the step of entering the index into the corresponding list of the fragment look-up table, includes newly entering the result of the

looked-up fragment ID and the index into the fragment look-up table, if the list corresponding to the result of the looked-up fragment ID does not exist in the fragment look-up table).

Consider **claim 3**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose the claimed method, wherein the step of transmitting the received packet includes attaching the index as the tag to a packet that has been previously received and stored in a fragment buffer and transmitting the previously received and stored packet to the corresponding one of the protocol processors, if the received packet is the first fragment and the list corresponding to the result of the looked-up fragment exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry for a first fragment; Flowchart of Fig. 4, blocks 110-114; paragraphs 0039-0041 that disclose the process of generating an index for a primary fragment (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, so that the primary and the related secondary fragments may be translated and transmitted to their destination host/port); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing list as disclosed in

paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing a method wherein the step of transmitting the received packet includes attaching the index as the tag to a packet that has been previously received and stored in a fragment buffer and transmitting the previously received and stored packet to the corresponding one of the protocol processors, if the received packet is the first fragment and the list corresponding to the result of the looked-up fragment exists in the fragment look-up table).

Consider **claim 4**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose the claimed method, wherein if the received packet is not the first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108, 118, 120; paragraph 0042, lines 1-4 that disclose the processing of a received packet that is not the first fragment), further comprising the steps of:

looking-up the fragment ID of the received packet and comparing the result of the looked-up fragment ID with each list of the fragment look-up table, to determine if there is a corresponding list; entering the result of the fragment ID looked-up for the received packet into the fragment look-up table, if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table; and storing the received packet in a fragment buffer (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the

arrival of the packet; this is achieved by checking packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing a method wherein if the received packet is not the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list, and entering the result of the fragment ID looked-up for the received packet into the fragment look-up table, if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table; and storing the received packet in a fragment buffer);

Consider **claim 5**, and **as it applies to claim 4 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., show and disclose a method of the claimed invention, wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet), further comprising the steps of: determining whether the index corresponding to the result of the tunnel ID look-up exists in the corresponding list (in Ganesan et al. reference, Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the

tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing looking up a tunnel ID of the received packet from a tunnel ID look-up table); and

attaching the index as the tag to the received packet and transmitting the received packet to the corresponding one of the protocol processors, if the index exists in the corresponding list (in Crow et al. reference, Fig. 4, block 116, paragraph 0039 that discloses the translation and subsequent transmission process for the primary fragment; Flowchart of Fig. 4, block 124; paragraph 0043, lines 7-13 which disclose that the secondary fragment is translated using the identified entry 90 and transmitted to one of the protocol processors); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing list as disclosed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table, so as to be able to attach the index as a tag to the received packet and transmit the received packet to the corresponding one of the protocol processors, if the index exists in the corresponding list).

Consider **claim 6**, and **as it applies to claim 5 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose the claimed method, comprising the step of storing the received packet in the fragment buffer, if the

index does not exist in the corresponding list (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists; the value of the current tail link for the queue is modified to point to a newly allocated block in data memory to store the received packet); and (in Crow et al. reference, Flowchart of Fig. 4, blocks 108 and 120; paragraph 0042, lines 14-17 which disclose that the secondary fragment 34 is stored in the fragment memory 84, if a fragment-context 92 does not exist for the secondary fragment 34 in the translation table 82).

Consider **claim 7**, Crow et al. show and disclose an apparatus for distributing a plurality of packets to a plurality of protocol processors (Fig.1, that shows a Border Router 16 receiving a plurality of packets from the Internet 22, and distributing the packets to a plurality of protocol processors 24; paragraph 0018 discloses the same details) comprising:

a dependant interface for transmitting the packet attached with the index to the corresponding one of the protocol processors (In Fig. 1, shown by the left link 20 between the Border Router 16 and Protocol Processor Hosts 24; Fig. 3 that shows a fragment context (an index) attached to the translated packet entry for transmission to its destination host/port).

However, Crow et al. do not explicitly disclose an apparatus comprising a receiving unit for receiving the packets from a network; a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment looked-up on

the received packet is entered; a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists; a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively; and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet.

In the same field of endeavor, Rana et al. disclose an apparatus comprising a receiving unit for receiving the packets from a network (Fig. 1, Input 12 and Input Interface 14 forming a receiving unit for receiving the packets from a network; paragraph 0019 that discloses the same details).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide an apparatus comprising a receiving unit for receiving the packets from a network, as taught by Rana et al., in the apparatus of Crow et al., so that the input packets can be received by the apparatus.

However, Crow et al., as modified by Rana et al., do not explicitly disclose an apparatus comprising a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment looked-up on the received packet is entered; a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists; a tunnel ID look-up table storage unit for storing a

tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively; and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet.

In the same field of endeavor, Varma et al. disclose an apparatus comprising a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment looked-up on the received packet is entered (Fig. 3, Control Memory 200 used for storing fragment look-up table with a plurality of lists within it, each list represented by a head pointer, a tail pointer and a packet count value, the lists storing fragment id (queue id) value of received fragmented packets stored in the data memory 210 and referenced by the link memory 220); and a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists (paragraph 0032 which discloses a queue processor that creates a new list and stores the received packet data each time a first new packet is received and for every subsequent packet received, compares the queue id of the incoming packet with the queues already in the control memory to place the incoming packet in the existing queue).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide an apparatus comprising a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment

looked-up on the received packet is entered; a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists, as taught by Varma et al., in the apparatus of Crow et al., as modified by Rana et al., so that the related packets can be grouped in a common queue (list) for subsequent translation and transmission to their destination host/port.

However, Crow et al., as modified by Rana et al. and Varma et al., do not explicitly disclose an apparatus comprising a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively; and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet.

In the same field of endeavor, Ganesan et al. disclose an apparatus comprising a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively (Figure 6B, remote hash table rhashtbl_t, that includes storage for vpn_id which corresponds to tunnel information table); and
a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet (Fig. 8, VPN/IKE Module 830 performing the function of a tunnel ID look-up device; Flowchart of Fig. 11, blocks 1120 and 1122;

paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide an apparatus containing a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively, and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet, as taught by Ganesan et al., in the apparatus of Crow et al. as modified by Rana et al. and Varma et al., so that encapsulated received packets can be securely delivered through the firewall of the receiving node.

Consider **claim 8**, and **as it applies to claim 7 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus, wherein if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table, the fragment look-up device newly enters the result of the looked-up fragment and the index into the fragment look-up table, if the received packet is a first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108-116; paragraph 0039 that discloses the processing of a received packet that is the

first fragment); (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the data; by checking the packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thereby disclosing creating a new list with the fragment id (queue id), if the list corresponding to the result of the looked-up fragment ID does not exist in the fragment look-up table); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry; Flowchart of Fig. 4, blocks 110-114; paragraph 0040 that discloses the process of generating an index (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, thereby disclosing generating an index); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a new fragment, in which case a fragment id (index) is assigned to the data packet and stored in the link list); thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table, the fragment look-up device newly enters the result of the looked-up fragment and the index into the fragment look-up table, if the received packet is a first fragment); and

newly enters the result of the looked-up fragment into the fragment look-up table, if the received packet is not the first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108, 118, 120; paragraph 0042, lines 1-4 that disclose the processing of a

received packet that is not the first fragment); (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the packet; this is achieved by checking packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table, the fragment look-up device newly enters the result of the looked-up fragment into the fragment look-up table, if the received packet is not the first fragment).

Consider **claim 9**, and **as it applies to claim 7 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus comprising, if the list corresponding to the result of the looked-up fragment and including the index does not exist in the fragment look-up table, a fragment buffer for storing the received packet if the received packet is not the first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108, 118, 120; paragraph 0042 that discloses the processing of a received packet that is not the first fragment); (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the packet; this is achieved by checking packet count value for zero. If the count value is zero, the head pointer and the tail pointer are

both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table, the fragment look-up device stores the received packet in a fragment buffer, if the received packet is not the first fragment).

Consider **claim 10**, and **as it applies to claim 9 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus wherein if the list corresponding to the result of the looked-up fragment and including the index exists in the fragment look-up table, the fragment look-up device attaches the index as the tag to the received packet to transmit the received packet to the corresponding one of the protocol processors (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry for a first fragment; Flowchart of Fig. 4, blocks 110-114; paragraphs 0039-0043 that disclose the process of generating an index for a primary fragment (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, so that the primary and the related secondary fragments may be translated and transmitted to their destination host/port); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e.

belonging to an existing list as disclosed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment and including the index exists in the fragment look-up table, the fragment look-up device attaches the index as the tag to the received packet to transmit the received packet to the corresponding one of the protocol processors).

Consider **claim 11**, and **as it applies to claim 9 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus wherein in the case of the received packet being the first fragment, the fragment look-up device attaches the index as the tag to each packet being a subsequent fragment following the first fragment and being stored in the fragment buffer to transmit each subsequent fragment packet via the dependant interface to the corresponding one of the protocol processors, if the list conforming to the result of the looked-up fragment exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry for a first fragment; Flowchart of Fig. 4, blocks 110-114; paragraphs 0039-0041 that disclose the process of generating an index for a primary fragment (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, so that the primary and the related secondary fragments may be translated

and transmitted to their destination host/port); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing list as disclosed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing an apparatus wherein in the case of the received packet being the first fragment, the fragment look-up device attaches the index as the tag to each packet being a subsequent fragment following the first fragment and being stored in the fragment buffer to transmit each subsequent fragment packet via the dependant interface to the corresponding one of the protocol processors, if the list conforming to the result of the looked-up fragment exists in the fragment look-up table).

Consider **claim 12**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose a method wherein the other received fragment packets are stored in a fragment buffer (in Crow et al. reference, Fig. 1, Fragment Memory 84 that stores the other received fragment packets; paragraph 0032 which discloses a queue or a suitable memory data structure for storing one or more secondary fragments),

wherein a list is stored in the fragment look-up table for each of fragmented packets (in Varma et al. reference, Fig. 3; that shows Data Memory 210 for storing packets, Control Memory 200 for storing the fragment look-up table with head and tail pointers and other data (including packet count) for each list, and Link Memory 220 with pointers to keep track of the related packets of the list in the data memory; paragraph 0032, lines 6-13

which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the packet; by checking the packet count value for zero. If the packet count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing a method wherein if the received packet is the first fragment packet, looking-up a fragment ID (queue id) of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list);

wherein the fragment look-up table is stored separately from the fragment memory (in Varma et al. reference, Fig. 3; that shows Data Memory 210 for storing fragments, separate from the Control Memory 200 for storing the fragment look-up table);

wherein, if the ***received fragment packet is determined to be the*** first fragment packet, searching the look-up table for a first list that corresponds to the other received fragment packets, wherein the other received fragment packets together with the first fragment packet form a datagram (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received first fragment, all the secondary fragments previously received and stored in the data memory along with the currently received first fragment form a complete datagram).

Consider **claim 13**, and **as it applies to claim 12 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose a method wherein, if the ***received fragment packet is determined to be the*** first fragment packet (in Crow et al. reference, Flowchart of Fig. 4, blocks 108-116; paragraph 0039 that discloses the processing of a received packet that is the first fragment), ***and*** the first list is found in the look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet), editing the list to update the index and searching the fragment buffer for the other received fragment packets and transmitting the found other received fragments based on the updated index of the first list ***without assembling the fragment packets to form a datagram*** (in Crow et al. reference, Fig. 3, Fragment Context field 92, an index updated in the Address Translation Entry field 90 after the first fragment packet is received; paragraphs 0039-0041 that disclose generating a fragment-context 92 for the identified translation entry 90 and associating the corresponding fragment-contexts of the secondary entries in the first list with that of the first fragment packet, so that the other (secondary) fragment packets received prior to the first fragment packet can be appropriately translated and transmitted to their destination as well**; in Varma et al. reference, the use of linked-lists shown in Fig. 3 and paragraph 0032 discloses transmitting the received fragments without assembling the fragment packets to form a datagram**).

Consider **claim 14**, and **as it applies to claim 13 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., disclose the claimed method, wherein, if ***the received fragment packet is determined to be*** one of the other fragment packets, searching the look-up table for the first list, and if the first list is not present, generating the first list comprising source address, destination address and an index and storing the one of the other fragment packets in the fragment buffer (in Rana et al. reference, Fig. 3c that shows the data structure for a link list memory comprising a session id (SID) field; paragraph 0037 which discloses that the link list memory 24 (shown in Fig. 1) is used to associate the blocks in packet memory 20 that form the PDUs (Packet Data Units) from the same session or traffic flow; paragraph 0020 which discloses that the link lists in the link list memory 24 are used by the queue engine 10 to track pointers associated with data packets stored in packet memory 20; paragraphs 0021-0023 which further disclose that a session could be identified and assigned a session id based upon the source address, destination address and any other field or combination of fields from the header of the data packet which form a unique identifier (corresponding to the first list); further disclosing that if the data packet is a fragment from a known session (existing first list), a fragment id is associated with the data packet, or if the data packet is a new fragment (non-existing first list), a fragment id based on the session id is assigned to the data packet).

Consider **claim 15**, and **as it applies to claim 14 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further disclose the claimed method, wherein the fragment look-up table further comprises a field indicating storage location of respective at least one other fragment packet in the fragment buffer (in Rana et al. reference, Fig. 3c, field marked "Next" that indicates storage location of respective at least one other fragment packet in the fragment buffer; paragraph 0037 that discloses the same details).

Consider **claim 16**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose a method wherein, if the received packet is the first fragment packet (in Crow et al. reference, Flowchart of Fig. 4, blocks 108-116; paragraph 0039 that discloses the processing of a received packet that is the first fragment),
searching an index indicating one of the protocol processors (in Crow et al. reference, Fig. 3, that shows a searched Address Translation Entry field 90 in the Address Translation Table 82 of Fig. 1, that matches the protocol processor and destination port in the IP and Transport Headers of the first fragment shown in Fig. 2A); and
corresponding to the tunnel ID of the received packet from a tunnel ID look-up table (in Ganesan et al. reference, Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing looking up a tunnel ID of the received packet from a tunnel ID look-up table), and

if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet),

updating the index into the corresponding list of the fragment look-up table (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a new fragment, in which case a fragment id (index) is assigned to the data packet and stored in the link list); and

transmitting the other received fragment packets stored in a fragment buffer based on the updated index stored in the corresponding list of the fragment look up table (in Crow et al. reference, Fig. 3, Fragment Context field 92, an index updated in the Address Translation Entry field 90 after the first fragment packet is received; paragraphs 0039-0041 that disclose generating a fragment-context 92 for the identified translation entry 90 and associating the corresponding fragment-contexts of the secondary entries with that of the first fragment packet, so that the other (secondary) fragment packets received prior to the first fragment packet can be appropriately translated and transmitted to their destination as well).

Response to Arguments

Applicant's arguments filed 05/02/2008 have been fully considered but they are not persuasive. The examiner respectfully disagrees with the applicant's arguments as the applied reference(s) provide more than adequate support and clarification. The examiner's rejection of 02/11/2008 is therefore maintained and made final. The examiner's response to the presented arguments is as follows:

Consider independent **claim 1.** The applicant alleges that the cited Varma et al. reference (US Patent Application Publication # 2004/0037302 A1) fails to disclose or suggest that "if the data is first data, searching to determine if there is a corresponding list". The examiner respectfully disagrees. In the Varma et al. reference, the Control Memory 200 (Fig. 3) is searched to determine if the fragment count (part of the other fields) is zero, representing an empty list initially. If the count is set to zero, the corresponding head and tail pointers (also set to null initially) in the Control Memory are set to the address in the Data Memory 210 (e.g. 21 in Fig. 3) where the first received fragment (not necessarily the first fragment of the datagram) is saved. Also the matching entry (e.g. 21) in the Link Memory 220 is set to null or empty to indicate that there is no next fragment in the linked list yet. The fragment count in the Other Fields of the Control Memory is incremented by 1.

When the next fragment is received, the corresponding list 15 in the Control Memory is searched again for the fragment count value (now set to 1). The second received fragment is saved in the next available location in the Data Memory (e.g. at address 34, because the intervening addresses may have been filled by fragments from

other datagrams), and the Link Memory address of the first received fragment (at entry 21) is changed from null to 34 to indicate where in the Data Memory the second fragment is stored. Also the corresponding address (34) in the Link Memory is set to null to indicate that there is no other fragment after the second received fragment. In the Control Memory, the Tail pointer is set to 34, and the fragment count is increased by 1 to a value 2. This process continues, until the next received fragment is the first fragment of the datagram. The Head pointer in the Control Memory is thereupon updated to point to a location in the Data Memory where the first datagram fragment is saved. The fragment count field and the Link Memory pointers are appropriately updated as each datagram fragment is received to form a fragment sequence ordered linked list. Finally, the Tunnel ID in the first datagram fragment is converted to a corresponding index of the protocol processor, as described in the Ganesan et al. reference (US Patent Application Publication # 2003/0069973 A1). The received datagram fragments can then be transmitted without reassembly.

Therefore, it is clear that the Varma et al. reference does require searching a corresponding list in order to update various pointer values for the linked list and maintain the fragment count.

The applicants further allege that in Varma et al. reference, there is no disclosure or even remote suggestion to look for other received packets. The examiner begs to differ. As explained above, as each fragment is received and processed, the fragment count and all pointer fields in the Control and Link Memory are correspondingly updated, without which the linked list will not operate. The fragment ID of each received

fragment has to be compared with the fragment ID in the Control Memory to be able to update corresponding fragment count and the pointers in the Link Memory. The examiner requests the applicant to carefully review the response above by creating three or more datagram fragments received out of order and place them at random locations in the Data Memory, then update the corresponding pointers in the Control and Link Memory for better understanding of the process described in the Varma et al. reference.

The applicant further alleges that nowhere in the cited Rana et al. reference (US Patent Application Publication # 2002/0095512 A1), is the claimed feature "if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering an index into the corresponding list of the fragment look-up table" disclosed. The examiner again respectfully disagrees with such allegation. First, the existence of the corresponding list is disclosed in the Varma et al. reference as described above. The Rana reference was used only to show and disclose entering an index into the corresponding list of the fragment look-up table. The Rana et al. reference not only handles packet fragments, but also handles out of order received packets of a traffic flow by associating a session id (corresponding to a linked list id for a set of fragments) with a list of packets, as disclosed in paragraph 0011. The value of the session id acts as an index to associate the data packet with a particular traffic flow. Paragraph 0022 in the Rana et al. reference discloses that the PDU assembler assigns a session id to the first data packet in a session; further disclosing that the session id is a location in session CAM 38, which is associated with the unique signature used to

identify each session, thereby disclosing entering an index into the corresponding list of the fragment look-up table that includes packets received out of sequence, then appropriately assembled in the corresponding linked lists (both fragment and packet linked lists). As a result, all the claimed features of claim1 are adequately disclosed by the four cited references, and therefore, **claim 1 is deemed non-patentable** by the examiner. The corresponding **dependent claims 2-6 and 12-16 are also rejected based on their dependency on the rejected claim 1**.

The applicant's argument that the dependent claim 12 further recites the limitation "if the received fragment packet is determined to be the first fragment packet, searching the look-up table for a first list that corresponds to the other received fragment packets, wherein the other received fragment packets together with the first fragment packet from a datagram", does not make the claim allowable, because the cited Varma et al. reference does disclose this claim feature.

Next, consider **independent claim 7 and its dependent claims 8-11**. There are no new arguments presented, so there is no response provided by the examiner.

In conclusion, the cited references, in combination, do adequately disclose all the features of **all 16 claims, none of which is therefore allowable.**

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any response to this Office Action should be **faxed to** (571) 273-8300 **or mailed to**:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Art Unit: 2143

Hand-delivered responses should be brought to

Customer Service Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Kishin G. Belani whose telephone number is (571) 270-1768. The Examiner can normally be reached on Monday-Friday from 6:00 am to 5:00 pm.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Nathan Flynn can be reached on (571) 272-1915. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free) or 703-305-3028.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist/customer service whose telephone number is (571) 272-0800.

/Kishin G Belani/
Examiner, Art Unit 2143

2008-07-30

/Nathan J. Flynn/
Supervisory Patent Examiner, Art Unit 2143